Journal of Computational and Applied Mathematics 227 (2009) 147–160

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Combining evolutionary and stochastic gradient techniques for system identification

Konstantinos Theofilatos [a], Grigorios Beligiannis [b,*], Spiridon Likothanassis [a]

[a] University of Patras, Department of Computer Engineering and Informatics, 26500 Rio, Greece

[b] University of Ioannina, Department of Business Administration of Food and Agricultural Enterprises, 30100 Agrinio, Greece

ARTICLE INFO

Article history:
Received 15 November 2007
Received in revised form 16 May 2008

Keywords:
System identification
Modeling
AutoRegressive Moving Average ARMA
Genetic Algorithms (GAs)
Least Mean Squares (LMS)
Hybrid evolutionary algorithms

ABSTRACT

In the present contribution, a novel method combining evolutionary and stochastic gradient techniques for system identification is presented. The method attempts to solve the AutoRegressive Moving Average (*ARMA*) system identification problem using a hybrid evolutionary algorithm which combines Genetic Algorithms (*GAs*) and the Least Mean Squares *LMS* algorithm. More precisely, *LMS* is used in the step of the evaluation of the fitness function in order to enhance the chromosomes produced by the *GA*. Experimental results demonstrate that the proposed method manages to identify unknown systems, even in cases with high additive noise. Furthermore, it is observed that, in most cases, the proposed method finds the correct order of the unknown system without using a lot of *a priori* information, compared to other system identification methods presented in the literature. So, the proposed hybrid evolutionary algorithm builds models that not only have small *MSE*, but also are very similar to the real systems. Except for that, all models derived from the proposed algorithm are stable.

1. Introduction

One of the most prominent issues in the field of signal processing is the adaptive filtering problem, with unknown time-invariant or time-varying parameters. Selecting the correct order and estimating the parameters of a system model is a fundamental issue in linear prediction and system identification. The problem of fitting an *ARMA* model to a given time series has attracted much attention because it arises in a large variety of applications, such as:

- Biology-Medicine (molecular modeling, cell modeling, etc.) [1]
- Computer networks (tail modeling) [2]
- Computer architecture (design, etc.) [3]
- Data bases (data modeling) [4]
- Mathematics [5]
- Meteorology (climatic modeling) [6]

A general model for *ARMA* can be represented as follows:

$$y(t) = \sum_{i=1}^{p} a_i \cdot y(t-i) + \sum_{j=1}^{q} b_j \cdot e(t-j) + e(t) \tag{1}$$

* Corresponding author.
E-mail addresses: theofilk@ceid.upatras.gr (K. Theofilatos), gbeligia@cc.uoi.gr, beligian@ceid.upatras.gr (G. Beligiannis), likothan@ceid.upatras.gr (S. Likothanassis).

0377-0427/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2008.07.014

where $y(t)$ is the observed time series data; $e(t)$ is a zero-mean white noise process with variance R, not necessarily Gaussian; $n = (p, q)$ is the order of the predictor; and a_i $(i = 1, \ldots, p)$, b_j $(j = 1, \ldots, q)$ are the predictor coefficients. Clearly the problem is two-fold: one has both to select the order of the predictor and then to compute the predictor coefficients. Let us define the vector of coefficients $\theta(t)$ as follows:

$$\theta(t) = (a_1(t)\ldots a_p(t) b_1(t)\ldots b_q(t))^{\mathrm{T}}. \tag{2}$$

The *ARMA* model identification problem is stated as follows: Given a set of time series observations $y(t)$, where $0 \leq t \leq N$, from an unknown *ARMA*(p, q) process we have to determine the unknown parameter vector:

$$v = [p\ q\ \theta(t)] \tag{3}$$

assuming that the order $n = (p, q)$ is unknown and the only available knowledge about the true order is that it satisfies the condition $n_0 \leq n \leq n_{\max}$. Various methods for linear-model order selection that represent information theoretical criteria exist. The best-known proposed solutions for this interesting problem include Akaike's information criterion (*AIC*), the final prediction error (*FPE*), and the minimum description length (*MDL*). Most of the techniques, computed by the above criteria, assume that the data follow Gaussian distribution, are based upon asymptotic results and are two-pass methods. Therefore, they cannot be used in an on-line or adaptive fashion. The *LMS* algorithm comprises a variation of the steepest descent algorithm that belongs to the widest category of stochastic gradient algorithms. We use the term *stochastic gradient* for the *LMS* algorithm to distinguish it from the steepest descent method. The basic difference between steepest descent and *LMS* is that the former uses the value of the inclination of the errors surface while the latter can be used to model the behavior of physical dynamic systems that are unknown (black boxes) and have one or more inputs and outputs. In the literature, *LMS* has been used in various applications. In [9], *LMS* is used in order to build an algorithm that corrects the errors obtained from the conversion of an analog signal to a digital one, while in [10], a modified version of *LMS* is used to process magnetocardiographies.

GAs are known to be one of the best methods for searching and optimization [7,8]. By applying genetic operators (reproduction, crossover, and mutation) in a population of individuals (sets of unknown parameters properly coded), they achieve the optimum value of the fitness function, which corresponds to the most suitable solution. As a result, they converge to the (near) optimal solution by evolving the best individuals in each generation. The main advantage of *GAs* is that they use the parameter values instead of the parameters themselves. In this way, they search the whole parameter space [11]. They have been applied to various searching and optimization problems as stated below:

- Analysis and prediction of data used in Biology and in Medicine (electroencephalography, magnetoencephalography, etc.) [12]
- Prediction of financial data (stock market prices, currency pars, etc.) [13]
- Business administration (scheduling, etc.) [14]
- Signal processing (modeling, time series prediction, etc.) [15]

GAs have been used extensively for the identification of unknown linear systems. They have been combined with many different classic optimization methods and as a result many very effective hybrid evolutionary algorithms have been created. One such hybrid evolutionary algorithm, which combines *GAs* with *RLS* is described in [16]. Also, in [17] an effective modeling method combining *GAs* with *Simulated Annealing* is presented.

The method presented in the current contribution comprises a method of *ARMA* system identification using a hybrid evolutionary algorithm. It combines GAs and the *LMS* algorithm. *LMS* is used in the step of the evaluation of the fitness function in order to enhance the chromosomes produced by the *GA*. Furthermore, after the termination criterion of the *GA* is satisfied, *LMS* is applied one more time in order to optimize the final result. In this way the advantages of both algorithms are utilized. The main advantage of *GAs* is that they can be applied to optimization problems with huge search space and many local extrema, while the main advantage of *LMS* is that it can be used for exhaustive and effective local search. The rest of the paper is organized as follows. In Sections 2 and 3 the *LMS* algorithm and Genetic Algorithms (*GAs*) are described explicitly. Next, in Section 4 the proposed hybrid evolutionary algorithm is presented in detail. In Section 5 experimental results are presented in order to prove the significance and efficiency of the proposed technique. Finally, Section 6 summarizes the conclusions and suggests future applications and extensions of the proposed algorithm.

2. The LMS

The *LMS* algorithm has been exhaustively described and analyzed in the literature [18]. *LMS* is a linear adaptive filter algorithm which consists of two basic procedures.

(1) A filtering procedure that comprises:
   (a) the computation of the filter's output $y(n)$ which is produced by a set of parameters.
   (b) the creation of the estimation error $e(n)$ produced by the comparison of the real output $y(n)$ with the desired output $d(n)$.
(2) An adaptive procedure which automatically regulates the parameters of the filter using the estimation error.

```
Procedure GA{
        t = 0;
        InitializePopulation P(t);
        evaluate P(t);
        until (done){
                t = t + 1;
                ParentSelection P(t);
                recombine P(t);
                mutate P(t);
                evaluate P(t);
                survive P(t);
        }
}
```

Fig. 3.1. A simple Genetic Algorithm.

The combination of the above procedures creates a feedback loop around the *LMS* algorithm. First, there is a transversal filter, around which the *LMS* algorithm is structured. This is responsible for the filtering procedure. Then, a procedure is executed which attempts to adapt to the filter's parameters [18]. The operation of the *LMS* algorithm can be described by the following equations:

(1) Filter output: $y(n) = \hat{\mathbf{w}}^{t}(n)\mathbf{u}(n)$
(2) Estimation error: $e(n) = d(n) - y(n)$
(3) Adaptation of the parameters: $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mathbf{u}(n)e(n)$

where $d(n)$ is the desired output and $\mathbf{u}(n)$ is the system input which is uniformly distributed over a specific range. The first two equations define the estimation error $e(n)$, whose computation is based on the present estimation of the parameter vector $\hat{\mathbf{w}}(n)$. Also, the term $\mathbf{u}(n)e(n)$ in the third equation, represents the correction applied in the present estimation of the parameter vector $\hat{\mathbf{w}}(n)$. This procedure starts with an initial estimation $\hat{\mathbf{w}}(0)$.

3. Genetic Algorithms

Genetic Algorithms (*GAs*) gained much popularity in the 90's. As a result, plenty of books have been published which describe thoroughly their structure and operation [11]. *GAs* comprise a rapidly evolved region of artificial intelligence. They are general purpose searching algorithms, which are based on evolution principles observed in nature and are becoming more and more popular because of their ability to solve difficult problems. *GAs* were firstly introduced by Holland [19], who traditionally used the most dominating representation which is binary strings. However, recent applications of *GAs* have focused on different representations such as graphs, Neural Networks, Lisp-Expressions, ordered lists and vectors of real numbers.

GAs are being characterized by their simplicity, their elegance as strong search algorithms and their ability to locate relatively fast the good solutions in difficult problems with many dimensions. They are frequently used as optimization tools, even though some researchers focus on their adaptive and generalization abilities. They can be applied in various optimization problems such as design, computer games, stock-market, medicine, adaptive control, telecommunications, the Traveling Salesman Problem, etc. Specifically, they are useful when:

- The search space is big, large, complicated and not fully understandable.
- The knowledge about the search space is little or the specific knowledge is difficult to be encoded in order to minimize the search space.
- There is no available mathematical analysis.
- Traditional methods have failed to solve the specific problem.

The solution to a problem is called a *chromosome*. A chromosome is constituted by a set of *genes* which are simple parameters which need to be optimized. A *GA* creates an initial *population* (a set of chromosomes) and evaluates this population. Next, it evolves the population for some *generations* in order to find a satisfying solution for the examined problem. In Fig. 3.1, the structure of a simple *GA* is described. After the initialization phase, the parents are chosen using a probabilistic function based on the fitness of the members of the population. In other words, the single chromosomes with the relatively bigger fitness are more likely to be chosen as parents. *N* children are created from *N* parents through the procedure of *crossover*. These *N* children will be *mutated* and some of them will survive in the next population replacing *N* other chromosomes of the population of the current generation. The application of mutation just reverses some bits of the chromosomes with low probability.

Function *InitializePopulation P(t)* creates a set of chromosomes which constitute the initial population of the algorithm. Function *evaluate P(t)* evaluates the population *P(t)*. In other words it computes the fitness of every chromosome of the

population for the specific problem that is examined. Function *ParentSelection P(t)* selects which chromosomes are going to be recombined in order to make the population of the new generation. Function *recombine P(t)* executes the crossover of the chromosomes which have been chosen as parents from the previous function, while function *mutate P(t)* applies mutation to the chromosomes that are created by function *recombine P(t)*. Finally, *survive P(t)* chooses which chromosomes will participate in the population of the next generation.

4. The proposed hybrid evolutionary algorithm

4.1. Basic ideas

The method proposed in this contribution comprises a hybrid evolutionary algorithm, that is, an algorithm that combines evolutionary computing with local searches. There is a lot of work on this kind of algorithms and their application on many difficult problems has shown very promising results [25–27]. The hybrid evolutionary algorithm presented in the current contribution, which combines *GAs* and the *LMS* algorithm, is applied to the *ARMA* system identification problem. The problem of identifying an unknown *ARMA* model is an open problem and many solutions have been proposed. However, there is not any known algorithm able to identify effectively all kinds of *ARMA* models. Also, many traditional methods like *IIRLMS* [29, 30] have been applied to the same problem and the results obtained were not satisfactory. There are two main difficulties in this problem. The search space is huge and it comprises of many local optima. The problem of identifying an unknown *ARMA* system is related to the problem of predicting time series. If we designed an algorithm that could identify every unknown *ARMA* model fast and precisely, then the same algorithm could be used to reliably solve many problems of time series prediction.

The proposed method is inspired by [20]. According to this method gradient methods are used to initialize a *GA* which is then used to find the final solution. So, the *GA* is used for local search. However, *GAs*, as known, have better performance when the surface in which they search is huge and has many local optima, while gradient methods perform better when they are used for local search. In the present contribution, we decided to use a *GA* to initialize a gradient method (in our case the *LMS* algorithm), that is, the *GA* is used for global search while the *LMS* is used for local exploration.

The gradient method used by the proposed hybrid evolutionary algorithm is, as stated above, a simple *LMS*, which provides an *MA* estimation of the unknown system to be identified. The reason for choosing *LMS* as gradient method are the following:

- It is simple to implement.
- It is on-line.
- It converges fast.
- It has small computational complexity.

Next, using the algorithm of Hartmut Brandestein and Rolf Unbehauen [21], the *MA* model, computed by the *LMS*, is transformed to an *ARMA* model whose order is defined by the *GA*. Considering that an *IIR* system is the limit of an *FIR* system someone can conclude that, under certain assumptions, the presented algorithm is expected to give very satisfactory results. An alternative approach would be to use another stochastic gradient technique instead of using *LMS*. *IIRLMS* for example, provides also as output an *ARMA* model. However, if the *IIRLMS* is used, many problems in implementing this algorithm will arise. The chromosomes used by the *GA* are vectors of varying length. So, in a chromosome like the one used by the presented algorithm there is no obvious way to distinguish the numerators' from the denominators' coefficients if the *IIRLMS* is used. These implementation difficulties prevented us from using an *IIR* stochastic gradient technique.

Supposing that the unknown models to be identified have, without loss of generality, the same number of coefficients both on the numerator and on the denominator, the unknown models can be described as follows:

$$\hat{h} = \frac{b_0 + b_1 z^{-1} + \cdots + b_{n-1} z^{-n}}{1 + a_0 z^{-1} + \cdots + a_{n-1} z^{-n}} \tag{4}$$

where n is the order of the system. This assumption is not expected to reduce the performance and the flexibility of the system, because if a coefficient does not exist in the system to be identified, it is expected that the algorithm will assign a near zero value to it. Finding the system's order is a very difficult task comprising of many local optima. For this reason, a *GA* is used in order to solve it. For simplicity and in order to reduce the complexity of the algorithm, the order of the system is supposed to be within the interval [1,10], which indeed includes a great number of systems. Another problem, that has to be faced, is that the specific order of the *MA* system, which will be used for every model in order to take satisfactory results, is not known. This problem has been faced by using linked lists as chromosomes. In this way the chromosomes are of varying length and the *GA* can be used to solve the problem of finding the best order of the *MA* model. In cases that the unknown system to be identified is an *MA* one, the algorithm skips the step of transforming the *MA* model to an *ARMA* one.

4.2. Explicit description of the proposed hybrid evolutionary algorithm

The proposed method constitutes of three phases which are described explicitly below:

Phase 1:

In this phase, the algorithm creates the initial population of the *GA*. Every chromosome is a type List object of the C++ Genetic Algorithms' library *GAlib* [22]. This permits the use of chromosomes with varying length. The first node of each list contains an integer in the interval [1, 10] which corresponds to the order of the *ARMA* system. The other nodes contain doubles which correspond to the *MA* filters' parameters which are going to be used as the initial filter for the *LMS* algorithm. The number of parameters of the *MA* filter and consequently the length of the chromosome, is chosen to be a random number which belongs to the interval $[3n, 5n]$, where n is the order of the system. When the *GA* starts the evolution procedure, the length of the chromosome can change and might take values out of the interval $[3n, 5n]$. The proposed algorithm does not allow the length of a chromosome to be less than $n + 2$ because the algorithm of Hartmut Brandestein and Rolf Unbehauen cannot build an *ARMA* system with order n from an *MA* system with order less than $n+1$. During the initialization procedure, the values of the parameters of the *MA* filter are random doubles created using the random number generator of GAlib [22].

Phase 2:

In this phase, the *GA* is applied to the initial population created during the previous phase. The *GA* used is the simple Genetic Algorithm provided by GAlib (*GASimpleGA*). The algorithm does not use overlapping populations [7] and implements the parents' selection using the roulette wheel selection method [7]. As far as crossover is concerned, the *one point crossover* operator for the type List object [22] is used with crossover probability equal to 0.6. This crossover picks a site between nodes in each parent. It is the same as single point crossover on a resizable binary string genome. The site in the mother is not necessarily the same as the site in the father. When a crossover site is picked, it is between nodes of the list. We first copy the mother into the child (this deletes whatever contents were in the child originally). Then, we clone the father from the cross site to the end of the list. Then, we delete the tail of the child from the mother's cross site to the end of the list. Finally, we insert the clone at the end of the child [22]. Regarding mutation, the *flip mutation operator* for the type List object is used with mutation probability equal to 0.01. This mutation operator mutates a list by changing the value of nodes. Any node has a p_{mut} chance of getting its value changed. Flip mutator randomly picks elements in the list then sets the element to any of the alleles in the allele set for this list genome. This will work for any number of allele sets for a given list [22]. Finally, the presented *GA* uses the elitism technique [7], which assures that the best chromosome of each generation is included in the next generation's population.

To evaluate the chromosomes and find the best one, the proposed algorithm follows the procedure presented below:

(1) With initial values the ones located in every chromosome, the proposed algorithm applies the *LMS* algorithm in a data *window* of length equal to *window* elements. The role and meaning of the *window* parameter is described later in this section.
(2) From the previous step an *MA* filter is created.
(3) Using the algorithm of Hartmut Brandestein and Rolf Unbehauen the *MA* filter is transformed to an *ARMA* filter. Its order is specified by the value of the first element of the chromosome (first node of the list).
(4) The Mean Square Error (*MSE*) of the *ARMA* system for the specific data *window* is computed.
(5) The performance (fitness) of each chromosome is computed using the following equation as objective function for the *GA*:

$$fitness = \frac{1}{c + MSE}. \tag{5}$$

Parameter c has to do with the convergence of the GA to the optimum solution. Using other values, as well as the classical value of 1, resulted in very small differences among all possible solutions, a fact that made it very difficult for the GA to converge to the optimum. The optimum value of parameter c, which equals 0.1, has been estimated after exhaustive results that proved that this value is the best value to assist the GA to converge to the optimum solution. The application of the *GA* on the initial population, for a specific number of generations, during which the selection, crossover and mutation operators are applied, leads to more effective populations.

Phase 3:

In this phase the proposed algorithm follows the steps shown below:

(1) The *LMS* algorithm is applied to data of length d_{lms} with initial parameters the ones located in the best chromosome computed by the *GA*. The role and meaning of the d_{lms} parameter is described later in this section.
(2) From the previous step an *MA* filter is created.
(3) Using the algorithm of Hartmut Brandestein and Rolf Unbehauen the *MA* filter is transformed to an *ARMA* filter. Its order equals to the value of the first element of the chromosome (first node of the list).
(4) The mean square error (*MSE*) of the *ARMA* system for the specific data *window* is computed.

Table 1
The regulation parameters used by the proposed hybrid evolutionary algorithm

μ_1	The step size used by the *LMS* which is used by the *GA* to evaluate its chromosomes (*Phase* 2)
μ_2	The step size used by the *LMS* in *Phase* 3 of the algorithm
I_{pop}	The size of the initial population of the *GA*
window	The length of the data *window* in which we apply the *LMS* in order to estimate the fitness of every chromosome of the *GA*
P_c	The crossover probability of the *GA*
P_m	The mutation probability of the *GA*
d_{lms}	The length of the data in which the *LMS* of *Phase* 3 is applied
$GA_{iterations}$	The number of generations of the *GA*

Table 2
The results of the preprocessing phase concerning the crossover probability

Crossover probability	Mean MSE	Highest MSE	Lowest MSE	s.d. of MSE
0.05	0.734	0.923	0.684	0.012
0.10	0.625	0.844	0.489	0.018
0.15	0.679	0.879	0.584	0.016
0.20	0.618	0.793	0.524	0.015
0.25	0.587	0.642	0.489	0.017
0.30	0.412	0.562	0.345	0.012
0.35	0.286	0.418	0.189	0.014
0.40	0.100	0.301	0.086	0.012
0.45	0.079	0.179	0.050	0.012
0.50	0.048	0.102	0.032	0.012
0.55	0.010	0.085	0.007	0.012
0.60	0.002	0.003	0.001	0.0001
0.65	0.005	0.010	0.003	0.0007
0.70	0.012	0.035	0.009	0.001
0.75	0.045	0.093	0.034	0.010
0.80	0.082	0.124	0.056	0.011
0.85	0.093	0.156	0.089	0.012
0.90	0.128	0.226	0.097	0.013
0.95	0.176	0.284	0.139	0.015

The *ARMA* filter resulted by the above procedure is the final filter that the proposed algorithm proposes as the best solution. The inputs and outputs of the *ARMA* system to be identified are used by the *GA* and the *LMS* algorithm which is used after the *GA* (in *Phase* 3). Also, the *GA* uses an *LMS* to evaluate the chromosomes in its objective function (*Phase* 2). Both *LMS* algorithms used by the proposed hybrid evolutionary algorithm, use the algorithm of Hartmut Brandestein and Rolf Unbehauen to transform the *MA* filter to an *ARMA* filter. So, the *GA* interacts with an *LMS* algorithm in order to evaluate its chromosomes through its objective function. Then, it passes the best chromosome that it has found to another *LMS* in order to optimize it. In the above description, there are a lot of regulation parameters. These parameters are described in detail in Table 1.

The values of the regulation parameters (crossover and mutation probabilities, the window size, etc.), which were described above must be selected very carefully so as to optimize the performance of the proposed hybrid evolutionary algorithm. Due to the fact, that there are no obvious criteria for the definition of the algorithm's parameter values in all instances of the problem, we decided, in each case as a preprocessing phase, to execute the algorithm for 100 times for each different value of each regulation parameter (while keeping the others stable) and use the values obtained from the simulation results for the rest of the experiments. In the following tables, we present the results of the experiments made in the preprocessing phase that made us decide on the specific values for the crossover probability, the mutation probability and the *window* size. The method followed in order to define the values of the rest regulation parameters is similar. For each different value of each regulation parameter 100 experiments were conducted and the average value of the MSEs of each experiment was calculated. In Table 2 the results of the preprocessing phase concerning the crossover probability are presented. From this table it is depicted that the most effective value as far as crossover probability is concerned is 0.6. In Table 3 the results of the preprocessing phase concerning the mutation probability are presented. From this table it is depicted that the most effective value as far as mutation probability is concerned is 0.01.

In Table 4 the results of the preprocessing phase concerning the *window* size are presented. As expected, the value of the *window* size is a crucial factor that affects significantly the performance of the proposed algorithm. A big value of the *window* size offers smaller *MSE* but slows down the algorithm. A small value allows the algorithm to run faster but results to a bigger *MSE*. From Table 4 it is depicted that the most effective value as far as *window* size is concerned is 50.

Such a parameter analysis may seem to be not scientifically supported to some readers; however this is not the case when the task is to define the optimal parameter values of a GA. Throughout the GAs literature there is no standard acceptable procedure or statistical methodology to define the optimal parameter values of a GA in all cases (problems) [7,8]. The only acceptable procedure for the determination of a GAs parameters is the use of a meta-GA [31,32] to estimate the optimal parameter values of the GA when applied to a specific problem. In order to demonstrate that the values selected for the regulation parameters were the proper ones we used a meta-GA to estimate them [31,32]. We executed the meta-GA for 100

Table 3
The results of the preprocessing phase concerning the mutation probability

Mutation probability	Mean MSE	Highest MSE	Lowest MSE	s.d. of MSE
0.002	0.016	0.021	0.010	0.0009
0.004	0.011	0.019	0.009	0.0008
0.006	0.007	0.015	0.004	0.0007
0.008	0.004	0.009	0.003	0.0005
0.010	0.003	0.005	0.001	0.0001
0.012	0.009	0.015	0.005	0.0004
0.014	0.015	0.023	0.009	0.0006
0.016	0.027	0.036	0.012	0.0009
0.018	0.036	0.056	0.021	0.0009
0.020	0.039	0.068	0.023	0.010
0.030	0.046	0.079	0.031	0.012
0.040	0.048	0.081	0.032	0.015
0.060	0.057	0.088	0.041	0.017
0.080	0.063	0.094	0.047	0.018
0.100	0.075	0.099	0.051	0.019
0.200	0.123	0.178	0.089	0.020
0.300	0.358	0.456	0.284	0.020
0.400	0.526	0.721	0.389	0.020
0.500	0.692	0.823	0.456	0.021

Table 4
The results of the preprocessing phase concerning the *window* size

Window size	Mean MSE	Mean execution time (s)
5	0.043	30
10	0.031	100
20	0.022	200
30	0.015	350
40	0.008	500
50	0.006	550
60	0.006	650
70	0.006	770
80	0.005	950
90	0.005	1220
100	0.005	1320
120	0.004	1450
140	0.004	1500
160	0.004	1610
180	0.003	1800
200	0.003	1930
250	0.002	2120
300	0.002	2360
400	0.001	2710
500	0.001	3000

Table 5
The optimal values of the regulation parameters estimated by the meta-GA

Crossover probability	Mutation probability	Window size
0.596	0.099	49.98

times and the average value of the estimation of each regulation parameter was calculated. The results of these experiments concerning the optimal values of the crossover probability, the mutation probability and the window size are presented in Table 5. As presented in Table 5 the optimal values estimated by the meta-GA are really very close to the values obtained by the preprocessing phase of the proposed hybrid evolutionary algorithm.

5. Experimental results

In order to demonstrate the applicability and performance of the proposed hybrid evolutionary algorithm, several simulation examples were carried out. The performance of the proposed evolutionary algorithm was compared with five different algorithms presented in the literature and the results were very satisfactory.

Table 6
Results taken from the Classic *GA* and the *GA* with *Kaiadas*, both initialized by *LMS*, in their attempt to identify the system described by Eq. (6)

	GA initialized by LMS (MSE)		*GA* with *Kaiadas* initialized by LMS (MSE)	
	1st Generation	2000th Generation	1st Generation	2000th Generation
$n = 6, d = 20, P_m = 0.01$	0.4637	0.0452	0.4720	0.0446
$n = 6, d = 40, P_m = 0.01$	0.4637	0.0456	0.4720	0.0411
$n = 8, d = 20, P_m = 0.01$	0.1567	0.0402	0.1148	0.0425
$n = 8, d = 40, P_m = 0.01$	1.1567	0.0411	0.1148	0.0398
$n = 6, d = 20, P_m = 0.1$	0.4637	0.1656	0.4720	0.3303
$n = 6, d = 40, P_m = 0.1$	0.4637	0.1824	0.4720	0.3516
$n = 8, d = 20, P_m = 0.1$	0.1567	0.1565	0.1148	0.1148
$n = 8, d = 40, P_m = 0.1$	0.1567	0.1565	0.1148	0.1145

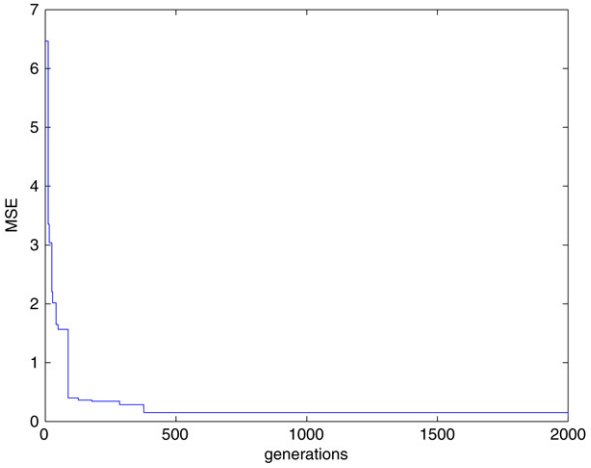

Fig. 5.1. The evolution of the *MSE* when the optimum parameter set of values is used (population size = 50, crossover probability = 0.6, mutation probability = 0.01)-Final MSE = 0.0381.

5.1. Example 1

The *ARMA* system to be identified is described by the following equation:

$$y(k) = -0.3y(k-1) + 0.4y(k-2) + 1.25u(k-1) - 2.5u(k-2) + n(k) \tag{6}$$

where $y(k)$ is system's output, $u(k)$ is system's input uniformly distributed in interval $[-2.5, 2.5]$ and $n(k)$ is additive noise uniformly distributed in interval $[-0.5, 0.5]$. The optimum values of the system coefficients are $A = [1, 0.3, -0.4]$, $B = [0, 1.25, -2.5]$, where a_i $(i = 1, 2, 3)$, b_j $(j = 1, 2, 3)$ are the *ARMA* system coefficients (see Section 1). In [20] Malavazos tried to identify the above *ARMA* system using a simple *GA* and a *GA* using the *Kaiadas* technique. Both algorithms were initialized by the *LMS* algorithm. We say that a *GA* uses the *Kaiadas* technique when the k chromosomes with the worst performance in each generation are not included in the next generation, where k is a parameter of the algorithm defined by the user. For a more detailed description of these algorithms refer to [20]. Results taken from these algorithms are shown implicitly in Table 6, where d is the number of the chromosomes that are chosen to be parents and n is the order of the model used.

In our experiments, we used a step value of 0.01 for the *LMS* used by the *GA* and a data *window* of length 100. For the *LMS* used after the *GA* we used a step value equal to 0.001 and a data *window* of length equal to 5000. The data *window* for the *LMS* used after the *GA* is so big because we want the *LMS* to explore and exploit the search space as effective as possible and find the (near) optimum solution. For every parameter set we executed the algorithm 100 times and the *MSE*s presented in Table 7 are the average of these executions. Fig. 5.1 shows the evolution of the *MSE* for the optimum parameter set of values. As stated before, this *optimum* parameter set of values was estimated in the preprocessing phase after exhaustive experiments.

Experimental results, presented in Table 8, demonstrate that the proposed hybrid evolutionary algorithm is more effective when a small mutation probability is used. On the other hand, a relative big crossover probability leads to slightly better results. Furthermore, when the size of the population of the *GA* is increased, an increment in the computational complexity of the algorithm is noticed, while the performance is not enhanced. The proposed hybrid evolutionary algorithm has many advantages compared to the method presented in [20]. To begin with, the method presented in [20], uses models of higher order than the real one and in this way the final proposed models are not similar to the real system. So, the proposed hybrid evolutionary algorithm finds the optimum model with the least number of coefficients. Also, the proposed method

Table 7
The *MSE* and the best *ARMA* coefficient values estimated for different sets of parameter values used by the *GA*

Parameters	Average MSE	Best numerator's coefficients computed	Best denominator's coefficients computed
$I_{pop} = 50, P_c = 0.4, P_m = 0.01$	0.1040	[0.092, 1.2493, −2.5059]	[1, −0.3020, −0.3866]
$I_{pop} = 50, P_c = 0.4, P_m = 0.1$	0.5014	[0.0295, 1.265, −2.5065]	[1, 0.3284, −0.3273]
$I_{pop} = 50, P_c = 0.6, P_m = 0.1$	0.3005	[0.0102, 1.2514, −2.5263]	[1, 0.3214, −0.3431]
$I_{pop} = 50, P_c = 0.6, P_m = 0.01$	0.0381	[0.0022, 1.2454, −2.4890]	[1, 0.3119, −0.3842]
$I_{pop} = 100, P_c = 0.2, P_m = 0.01$	0.1414	[−0.0348, 1.2531, −2.4996]	[1, 0.3104, −0.3727]
$I_{pop} = 100, P_c = 0.3, P_m = 0.01$	0.1106	[−0.0037, 1.2466, −2.5004]	[1, 0.3015, −0.3946]
$I_{pop} = 100, P_c = 0.2, P_m = 0.1$	0.6958	[0.0285, 1.2201, −2.4781]	[1, 0.3307, −0.3263]
$I_{pop} = 100, P_c = 0.3, P_m = 0.1$	0.3680	[−0.0414, 1.2928, −2.5639]	[1, 0.3310, −0.3094]

Table 8
Comparing experimental results of the proposed Hybrid Evolutionary Algorithm (HEA) with the algorithm presented in [20], both applied to the system described by Eq. (6)

Algorithm	Mean MSE	Highest MSE	Lowest MSE	S.d. of MSE
GA+LMS [20]	0.0402	0.0810	0.0372	0.021
Kaiadas [20]	0.0398	0.0720	0.0248	0.018
HEA	0.0381	0.0401	0.0219	0.008

Table 9
Comparing experimental results of the proposed Hybrid Evolutionary Algorithm (HEA) with the algorithms presented in [23], all applied to the system described by Eq. (7)

Algorithm	Mean MSE	Highest MSE	Lowest MSE	S.d. of MSE
ARMAX [23]	1.5740	1.8820	1.3300	0.323
OE RGO [23]	0.0388	0.0420	0.0300	0.0085
ARMAX RGO [23]	0.0163	0.0251	0.0100	0.0032
NARMAX RGO [23]	0.0034	0.0041	0.0030	0.0005
HEA	0.0029	0.0031	0.0026	0.0005

uses less *a priori* information concerning the system to be identified. Specifically, it uses as *a priori* information only the fact that the system's order lies in the interval [1, 10].

5.2. Example 2

In this example, the *ARMA* system to be identified is presented in [23] and is described by the following equation:

$$y(k) - 0.6y(k-1) + 0.4y(k-2) = u(k) - 0.3u(k-1) + 0.05u(k-1) + e(k) - 0.6e(k-1) + 0.4e(k-2) \quad (7)$$

where $y(k)$ is the system's output, $u(k)$ is the system's input which is uniformly distributed in interval $[-1, 1]$ and $e(k)$ is the noise which is uniformly distributed in interval $[-0.1, 0.1]$. In [23], four different algorithms are presented, all of which are variations of Restricted Genetic Optimization (*RGO*). *RGO* is a modified *GA* which uses semi-local optimization. For a more detailed description of these algorithms refer to [23]. The best *MSE* achieved by the algorithms presented in [23] is 0.003. In order to demonstrate the effectiveness of the proposed hybrid evolutionary algorithm we used it to identify the *ARMA* system of Eq. (7) using the optimum set of parameter values ($I_{pop} = 50, P_c = 0.4, P_m = 0.1$). The values for parameters μ_1 and μ_2 which are the steps of the two *LMS* algorithms used by the proposed hybrid evolutionary algorithm are 0.4 and 0.05, respectively.

The proposed algorithm was executed 100 times and the average *MSE* was estimated, which equals 0.0026. The system's coefficients estimated were $A = [1, -0.5876, 0.3668]$, $B = [1.0048, -0.2948, 0.0274]$. In Fig. 5.2 the evolution of the *MSE* is presented. Experimental results, presented in Table 9, demonstrate that the proposed hybrid evolutionary algorithm performs better than the four *RGO* algorithms presented in [23], while it uses less *a priori* information about the system to be identified. Specifically, the four *RGO* algorithms presented in [23] use as a fact that the exact order of the system to be identified is known *a priori*, while the proposed hybrid evolutionary algorithm use as a fact that the order of the unknown system belongs to interval [1, 10].

5.3. Example 3

In this example the *ARMA* system to be identified is presented in [24] and is described by the following equation:

$$y(k) = 0.6561y(k-4) + x(k) + 0.6x(k-1) - 0.3937x(k-3) + n(k) \quad (8)$$

where $y(k)$ is the output of the system, $x(k)$ is white Gaussian noise and $n(k)$ is additive noise. Both noises have zero mean value and unit dispersion.

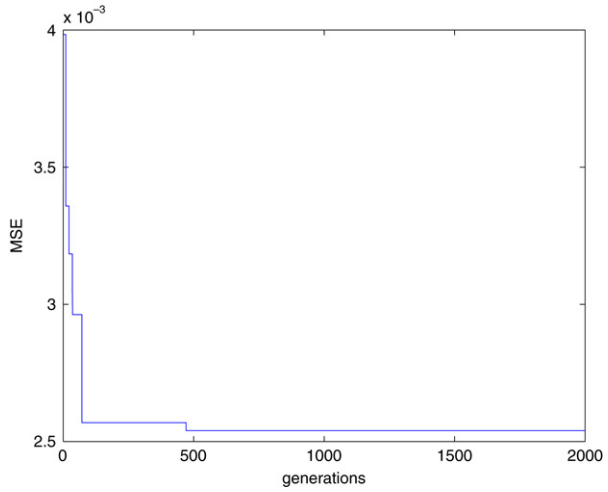

Fig. 5.2. The evolution of the *MSE* when the optimum set of parameter values is used (population size = 50, crossover probability = 0.6, mutation probability = 0.01)-Final *MSE* = 0.0026.

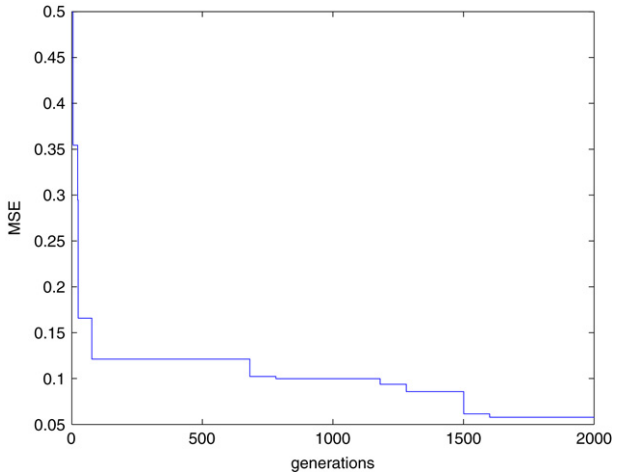

Fig. 5.3. The evolution of the *MSE* when the optimum set of parameter values (population size = 50, crossover probability = 0.6, mutation probability = 0.01) and no additive noise are used-Final *MSE* = 0.0607).

In order to demonstrate the effectiveness of the proposed hybrid evolutionary algorithm we tried to identify the system described in Eq. (8) using the proposed hybrid evolutionary algorithm. This system exhibits some special attributes. First of all, it has not the same order in the numerator and the denominator. Furthermore, some of its coefficients have zero values. Someone could expect that the proposed hybrid evolutionary algorithm would not be very efficient for the identification of this specific system. However, experimental results showed that this is not true. The set of parameter values used in all experiments is the optimum one ($I_{pop} = 50$, $P_c = 0.4$, $P_m = 0.1$). The values for parameters μ_1 and μ_2 which are the steps of the two *LMS* algorithms used by the proposed hybrid evolutionary algorithm are 0.05 and 0.005, respectively. Initially, we did not use any additive noise. The proposed algorithm was executed 100 times and the average *MSE* was estimated, which equals 0.0422. The system's coefficients estimated were $A = [1, -0.0495, 0.0138, 0.0244, -0.5369]$, $B = [1.0062, 0.5712, -0.0228, -0.3886, 0.1816]$. In Fig. 5.3 the evolution of the *MSE* is presented when no additive noise is used.

We repeated the previous experiment using, this time, additive noise with *SNR* = 20 dB. The average *MSE*, estimated after 100 runs, is 0.0767, while the system's coefficients were $A = [1, -0.4943, 0.0116, -0.0054, -0.5458]$, $B = [1, 0017, 0.5634, 0.0529, -0, 4283, 0.1621]$. This result demonstrates that the existence of noise does not affect the performance of the algorithm. In Fig. 5.4, the evolution of the *MSE* is presented when additive noise with *SNR* = 20 dB is used. In [24] no results concerning the identification of the above *ARMA* system are presented. However, we decided to present this example in order to demonstrate that the proposed hybrid evolutionary algorithm achieves satisfactory results even in cases where the systems to be identified have not the same order in the numerator and the denominator. This fact is demonstrated by the low *MSE* resulted for both unknown systems with or without the presence of additive noise.

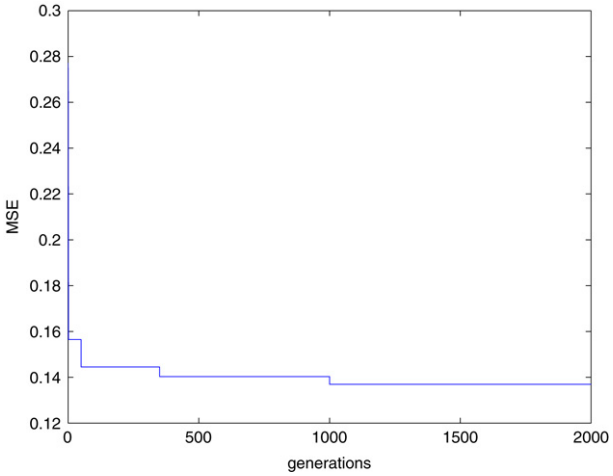

Fig. 5.4. The evolution of the *MSE* when the optimum set of parameter values (population size = 50, crossover probability = 0.6, mutation probability = 0.01) and additive noise of *SNR* = 20 dB are used-Final *MSE*=0.1487).

Table 10
The coefficients' values estimated by algorithms *FOS* and *OPS* in identifying the system described by Eq. (9)

Algorithms	Average *MSE*	Best numerator's coefficients computed	Best denominator's coefficients computed
OPS (0 dB)	0.08	[−0.54, 0.34, 0.34, 0.21, 0.00, 0.00]	[1, −0.35, 0.32, 0.10, 0.00]
FOS (0 dB)	0.09	[−0.54.0.40, 0.00, 0.34, 0.03, 0.06]	[1, −0.24, 0.00, 0.00, 0.11]
OPS (10 dB)	0.74	[−0.54, 0.54, −0.04, 0.26, −0.10, 0.08]	[1, 0.00, 0.18, 0.00, 0.00]
FOS (10 dB)	0.88	[−0.55, 0.47, 0.00, 0.28, 0.00, 0.04]	[1, −0.13, 0.12, 0.00, 0.09]

Table 11
Comparing experimental results of the proposed Hybrid Evolutionary Algorithm (HEA) with the algorithms presented in [28], all applied to the system described by Eq. (9)

Algorithm	Mean MSE	Highest MSE	Lowest MSE	S.d. of MSE
OPS (10 dB) [28]	0.7400	0.8324	0.6923	0.1471
FOS (10 dB) [28]	0.8800	0.9589	0.8230	0.1924
HEA (10 dB)	0.0850	0.0910	0.0743	0.0082
OPS (0 dB) [28]	0.0800	0.0920	0.0752	0.0032
FOS (0 dB) [28]	0.0900	0.1043	0.0864	0.0085
HEA (0 dB)	0.0048	0.0051	0.0044	0.0005

5.4. Example 4

In this example the ARMA system to be identified is presented in [28] and is described by the following equation:

$$y(n) = 0.35y(n-1) + 0.32y(n-2) + 0.1y(n-3) - 0.54x(n) + 0.34x(n-1) + 0.23x(n-2) + 0.21x(n-3) + v(n) \quad (9)$$

where $y(n)$ is the system's output, $x(n)$ is the system's input which is white Gaussian noise and $v(n)$ is additive noise. Both noises have zero mean value and unit dispersion. In [28] two algorithms are presented for system identification named *OPS* and *FOS*, respectively. For a more detailed description of these algorithms refer to [28]. Both algorithms need to know *a priori* the order of the system to be identified. These algorithms were tested in identifying the system described by Eq. (9) when the order of the system was chosen to be 5 (instead of 4 which is the real order) for the denominator and 6 (instead of 4 which is the real order) for the numerator [28]. Many experiments were conducted without additive noise and with additive noise of 10 dB. The values of the coefficients estimated are shown in Table 10 [28].

The optimum solution to this problem is $A = [1, -0.35, -0.32, -0.1]$, $B = [-0.54, 0.34, 0.23, 0.21]$. As shown in Table 10 the estimations given by OPS and FOS are not close enough to the real system. We attempted to identify the above system using the proposed hybrid evolutionary algorithm. The set of parameter values used is the optimum one ($I_{pop} = 50$, $P_c = 0.4$, $P_m = 0.1$). The step values of the two *LMS* algorithms used by the proposed hybrid evolutionary algorithm are 0.05 and 0.005, respectively. At first, we did not use any additive noise. The average *MSE* resulted after 100 runs was 0.0048 (see Table 11). The system's coefficients estimated were $A = [1, -0.3920, -0.3101, -0, 0610]$, $B = [-0.5480, 0.3610, 0.2194, 0.1874]$. In Fig. 5.5, the evolution of the *MSE* is presented when no additive noise is used.

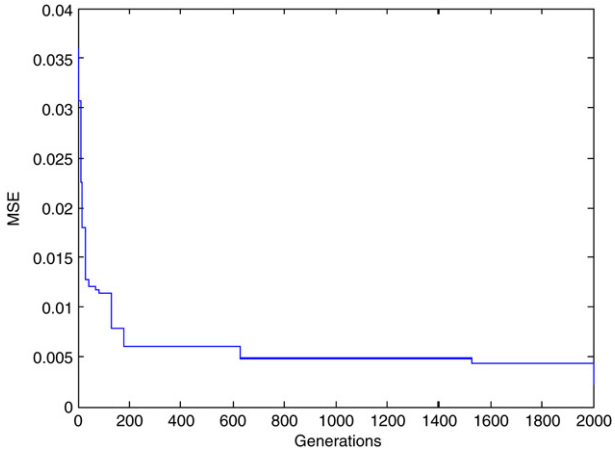

Fig. 5.5. The evolution of the *MSE* when the optimum set of parameter values (population size = 50, crossover probability = 0.6, mutation probability = 0.01) and zero additive noise are used-Final *MSE* = 0.0005).

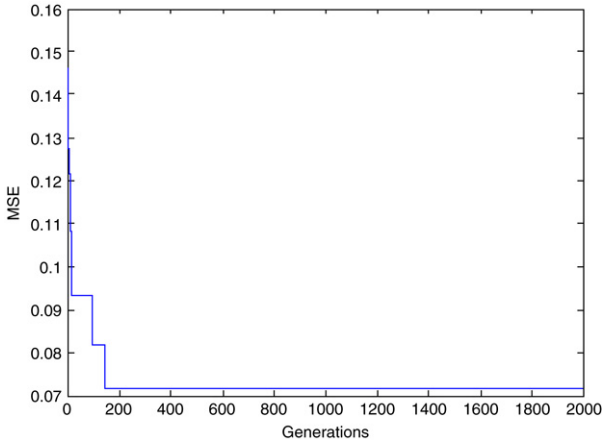

Fig. 5.6. The evolution of the *MSE* when the optimum set of parameter values (population size = 50, crossover probability = 0.6, mutation probability = 0.01) and additive noise with *SNR* = 10 dB are used-Final *MSE* = 0.072).

We repeated the previous experiment with presence of additive noise with $SNR = 10dB$. The average *MSE* resulted after 100 runs was 0.085 (see Table 11). The system's coefficients estimated were $A = [1, -0.3269, -0.1921, -0.1593]$, $B = [-0.5553, 0.3271, 0.1612, 0.2779]$. The average *MSE* estimated is bigger than before, due to the presence of additive noise. However, the performance of the proposed hybrid evolutionary algorithm is much better than the performance of algorithms *OPS* and *FOS* because the coefficient values estimated are much closer to the real ones compared to the coefficient values computed by algorithms *OPS* and *FOS*. In Fig. 5.6, the evolution of the *MSE* is presented when additive noise with $SNR = 10dB$ is used.

5.5. Example 5

In this example the system to be identified is an *MA* system, is presented in [24] and is described by the following equation:

$$y(n) = 0.34x(n) - 0.23x(n-1) + 0.50x(n-2) + 0.75x(n-3) - 0.15x(n-4) - 0.4x(n-5) + 0.35x(n-6) - 0.2x(n-7) + v(n) \quad (10)$$

where $y(n)$ is the system's output, $x(n)$ is the system's input which is white Gaussian noise and $v(n)$ is additive noise. Both noises have zero mean value and unit dispersion. It is supposed that we know *a priori* that the system is an *MA* one. We tried to identify the system described by Eq. (10) using the proposed hybrid evolutionary algorithm. The set of parameter values used in this example is the optimum one ($I_{pop} = 50, P_c = 0.4, P_m = 0.1$). The step values of the two *LMS* algorithms used by the proposed hybrid evolutionary algorithm are 0.05 and 0.005, respectively. At first, we did not use any additive noise. The average *MSE* resulted after 100 runs was practically 0. The *MA*

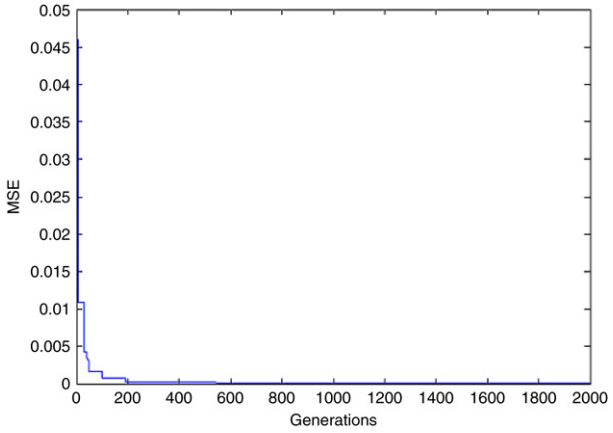

Fig. 5.7. The evolution of the *MSE* when the optimum set of parameter values (population size = 50, crossover probability = 0.6, mutation probability = 0.01) and zero additive noise are used-Final $MSE = 2.54638e-021$).

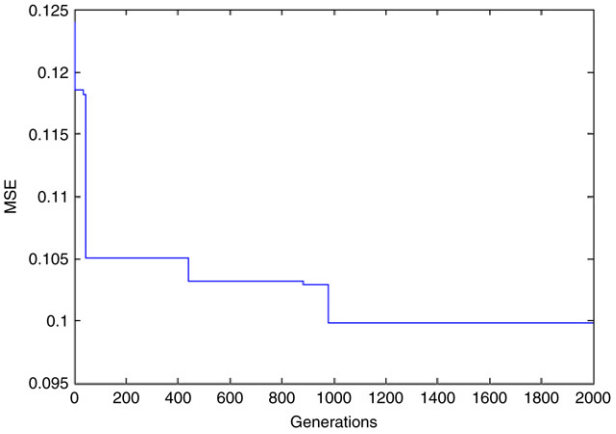

Fig. 5.8. The evolution of the *MSE* when the optimum set of parameter values (population size = 50, crossover probability = 0.6, mutation probability = 0.01) and additive noise with $SNR = 10$ dB are used-Final $MSE = 0.0975$).

system's coefficients estimated were $B = [0.34, -0.23, 0.5, 0.75, -0.15, -0.4, 0.35, 0.2]$. In Fig. 5.7, the evolution of the *MSE* is presented when no additive noise is used. We repeated the previous experiment using additive noise of $SNR = 10$dB. The average *MSE* that resulted after 100 runs was 0.0975. The *MA* system's coefficients estimated were $B = [0.33, -0.2250, 0.5017, 0.7667, -0.1712, -0.4177, 0.3674, 0.2054, 0.018]$. In Fig. 5.8, the evolution of the *MSE* is presented when additive noise with $SNR = 10dB$ is used. Experimental results obtained by the above example demonstrate the ability of the proposed hybrid evolutionary algorithm to identify *MA* systems reliably. In [24] no results concerning the identification of the above *MA* system are presented. However, we decided to present this example in order to demonstrate that the proposed hybrid evolutionary algorithm achieves satisfactory results even in cases where the systems to be identified are *MA* ones.

6. Conclusions and future work

From the examples presented above, it has been demonstrated that the proposed hybrid evolutionary algorithm is very effective in identifying unknown systems, even in cases with different order in the numerator and the denominator and the presence the high additive noise. Furthermore, we observed that in most cases, the proposed hybrid evolutionary algorithm found the correct order of the unknown systems using very little *a priori* information. So, the proposed hybrid evolutionary algorithm resulted in models that not only have small *MSE* but are also similar to the real system models. Except for that, all models derived from the proposed hybrid evolutionary algorithm are stable. The biggest disadvantage of the proposed algorithm is its relative big computational complexity, which comes as a result of using a *GA*.

Another very important issue of the proposed algorithm is the selection of the optimum set of the regulation parameter values. As stated before, due to the fact that there are no obvious criteria for the definition of the algorithm's parameter values in all instances of the problem, we decided, in each case as a preprocessing phase, to execute the algorithm for 100 times for each different value of each regulation parameter (while keeping the others stable) and use the values obtained from

the simulation results for the rest of the experiments. Two special regulation parameters whose values should be chosen correctly, in order the algorithm to result in satisfactory results, are the steps used by the two *LMS* algorithms used. In the present contribution, the values of these parameters were also chosen after exhausting tests. In each example presented, we provide both the step value of the *LMS* algorithm which runs inside the *GA* and the step value of the *LMS* algorithm which runs after the *GA*. We observed that the algorithm works better when the step value chosen for the *LMS* which runs inside the *GA* is much bigger than the step value for the *LMS* which runs after the *GA*. This is because, after the *GA* has completed its execution, we have to search locally in the neighborhood of the solution found by the *GA* in the most effective way. Furthermore, we observed that the optimum step values for both *LMS* algorithms depend on the type of the system's input (white noise, white Gaussian noise, uniform distribution etc.). As far as future work is concerned, we are thinking of using in the *GA*'s chromosomes some extra genes which will be used in order to estimate the optimum step values of the *LMS* algorithms. By using extra genes for the step parameters we assign the *GA* with the responsibility to find the optimum values for these parameters, too.

References

[1] B. Ribba, T. Colin, S. Schnell, A multiscale model for cancer and its use in analyzing irradiation therapies, Theoretical Biology and Medical Modeling 3 (7) (2006).
[2] R. Puigjaner, Performance modeling of computer networks, in: Proc. of the 2003 IFIP/ACM Latin America conference on Towards a Latin American agenda for network research, ACM, 2003, pp. 106–123.
[3] A. Paar, J. Reuter, J. Schaeffer, A pluggable architectural model and a formally specified programming language independent API for an ontological knowledge base server, in: Proc. of the 2005 Australasian Ontology Workshop, 58, 2005, pp. 83–91.
[4] H. Jin, M.-L. Wong, K.S. Leung, Scalable model-based clustering for large databases based on data summarization, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (11) (2005) 1710–1719.
[5] L. Rudolph, Mathematics, Models and Metaphors, Culture and Psychology 12 (2) (2006) 245–259, SAGE Publications.
[6] R.J. Yang, A.G. Xia, D.V. Michelangeli, D.A. Plummer, L. Neary, J.W. Kaminski, J. McConnell, Evaluating a Canadian regional air quality model using ground-based observations in north–eastern Canada and United States, The Royal Society of Chemistry, Journal of Environmental Monitoring 5 (2003) 40–46.
[7] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed., Springer-Verlag, NY, 1996.
[8] M. Mitchell, An Introduction to Genetic Algorithms (Complex Adaptive Systems), A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England, 1998.
[9] Y. Chiu, C.W. Tsang, B. Nikolie, P.R. Gray, Least mean square adaptive digital background calibration of pipelined analog to digital converters, IEEE Transactions on Circuits and Systems 51 (1) (2004) 38–46.
[10] X.M. Zhu, Y. Tian, S.P. Zhao, G.H. Chen, Q.S. Yang, A noise feedback least-mean-square algorithm of data processing for SQUID-based magnetocardiography, Institute of Physics Publishing Superconductor Science and Technology 18 (2005) 1054–1059.
[11] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, Mass, 1989.
[12] G.N. Beligiannis, L.V. Skarlas, S.D. Likothanassis, A generic applied evolutionary hybrid technique for adaptive system modeling and information mining, in: 'Signal Processing for Mining Information', IEEE Signal Processing Magazine 21 (3) (2004) 28–38 (special issue).
[13] J. Li, E.P.K. Tsang, Improving technical analysis predictions: An application of genetic programming, in: Proc. of the 12th International FLAIRS Conference, FLAIRS-99, USA, 1999, pp. 108–112.
[14] T. Terano, K. Naitoh, Agent-based modeling for competing firms: From balanced-scorecards to multi-objective strategies, in: Proc. of the 37th Hawaii International Conference on System Sciences-2004, 2004, pp. 1–8.
[15] J.V. Hansen, J.B. Mcdonald, R.D. Nelson, Time series prediction with genetic-algorithm designed neural networks: An empirical comparison with modern statistical models, Computational Intelligence 15 (3) (1999).
[16] K. Warwick, Y.-H. Kang, R.J. Mitchell, Genetic least squares for system identification, Soft Computing 3 (1999) 200–205.
[17] K.C. Tan, Y. Li, D.J. Murray-Smith, K.C. Sharman, System identification and linearization using genetic algorithms with simulated annealing, in: Proc. of the IEE/IEEE Int. Conf. on GA in Eng. Syst.: Innovations and Appl., 1995, pp. 164–169.
[18] S. Haykin, B. Widrow, Least-Mean-Square Adaptive Filters (Adaptive and Learning Systems for Signal Processing, Communications and Control Series), Wiley-Interscience, 2003.
[19] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, second ed., University of Michigan Press, 1992, MIT Press.
[20] K. Malavazos, Genetic algorithms and the problem of ARMA systems' identification, Degree project for Master studies in the Department of Computer Engineering and Informatics, University of Patras, Greece, 2002.
[21] H. Brandestein, R. Unbehauen, Least-squares approximation of FIR by IIR digital filters, IEEE Transactions on Signal Processing 46 (1) (1998).
[22] M. Wall, GAlib-A C++ Library of Genetic Algorithm Components, Massachusetts Institute of Technology (MIT), http://lancet.mit.edu/ga/.
[23] S. Garrido, L. Moreno, Learning adaptive parameters with restricted genetic optimization method, in: J. Mira, A. Prieto (Eds.), in: Lecture Notes in Computer Science, vol. 2084, Springer, Berlin, 2001, pp. 612–620.
[24] J. Mari, P. Stoica, T. McKenveley, Vector ARMA estimation: A reliable subspace approach, IEEE Transactions on Signal Processing 48 (7) (2000).
[25] G. Crina, A. Ajith, I. Hisao (Eds.), Hybrid Evolutionary Algorithms, in: Studies in Computational Intelligence, vol. 75, 2007.
[26] R. Salomon, P. Eggenberger, Adaptation on the evolutionary time scale: A working hypothesis and basic experiments, Artificial Evolution (1997) 251–262.
[27] G.D. Magoulas, V.P. Plagianakos, M.N. Vrahatis, Neural network-based colonoscopic diagnosis using on-line learning and differential evolution, Applied Soft Computing 4 (4) (2004) 369–379.
[28] S. Lu, K.H. Ju, K.H. Chon, A new algorithm for linear and nonlinear ARMA parameter estimation using affine geometry, IEEE Transactions on Biomedical Engineering 48 (10) (2001).
[29] I.D. Landau, Unbiased recursive identification using model reference techniques, IEEE Transactions on Automatic Control 21 (1976) 194–202.
[30] P.L. Feintuch, An adaptive recursive LMS filter, in: Proc. of the IEEE, 1976, pp. 1622–1624.
[31] J.E. Smith, T.C. Fogarty, Operator and parameter adaptation in genetic algorithms, Soft Computing 1 (2) (1997) 81–87.
[32] J. Clune, S. Goings, B. Punch, E. Goodman, Investigations in meta-GAs: Panaceas or pipe dreams? in: Proc. of the 2005 workshops on Genetic and Evolutionary Computation, Genetic and Evolutionary Computation Conference, 2005, pp. 235–241.